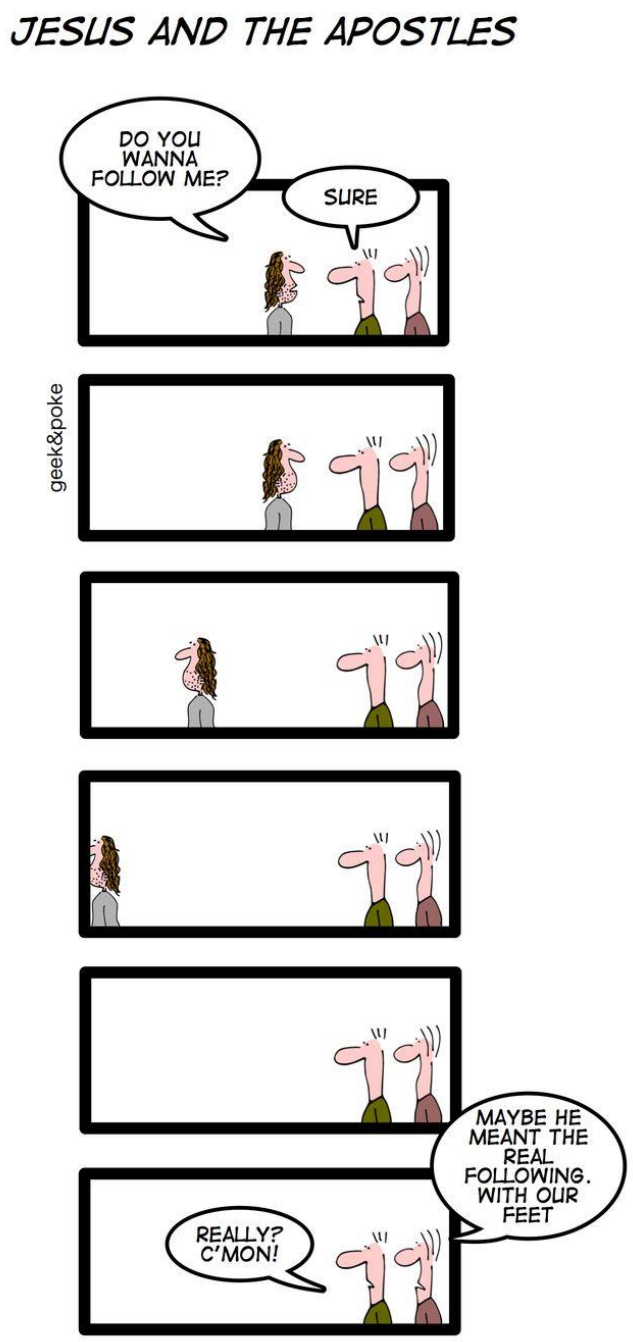
Les tableaux

Antoine SCHMID, CFPT-Informatique
Avec la collaboration du CPLN

Table des Matières

- Résumé de l’épisode précédent

- Les tableaux
  - Rôle
  - Créer un tableau
  - Accéder aux éléments
  - Parcourir un tableau
  - Ajouter des éléments
  - Supprimer des éléments



Dans l’épisode précédent…

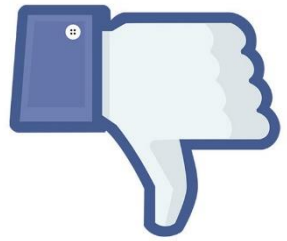
```
const heros = {
    nom = "Aurora",
    sante = 150,
    force = 25,

    // Renvoie la description du personnage
    decrire() {
      return `${this.nom} a ${this.sante} points de vie`;
    }
  };
```

Le rôle des tableaux

- Imaginez que vous souhaitiez informatiser la liste de tous les films que vous avez vus récemment.
- Une première solution serait de créer une variable par film

```
let film1 = "Django Unchained";
let film2 = "Interstellar";
let film3 = "Green Book : Sur les routes du sud";
...
```

- Un très grand nombre de variables
- Il n’est pas possible de faire des recherches, du tri..

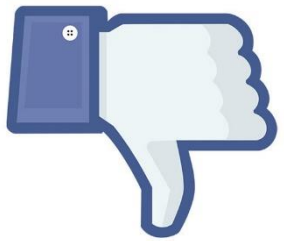


Le rôle des tableaux

- On pourrait stocker tous les titres dans une unique chaîne de caractères
- En choisissant un caractère pour délimiter les titres.

```
let films = "Django Unchained - Interstellar - Green Book - ...";
```

- Cette chaîne risque de devenir exagérément longue
- Et que faire si le caractère délimiteur est également présent dans le titre d'un des films

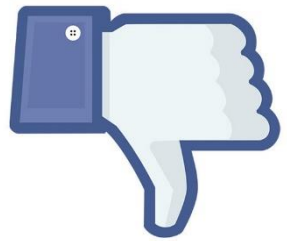


Le rôle des tableaux

- Une autre possibilité consiste à regrouper les films dans un objet

```
let films = {
  film1: "Django Unchained",
  film2: "Interstellar",
  film3: "Green Book : Sur les routes du sud",
  // ...
};
```

- les noms de ses propriétés sont inutiles et répétitifs
- A chaque nouveau film vu, il faudra ajouter à l'objet une propriété filmN sans se tromper sur la valeur de N



Solution : les tableaux !

- Un tableau est un type de donnée qui permet de stocker un ensemble d'éléments.

```
let films = [
  "Django Unchained",
  "Interstellar",
  "Green Book : Sur les routes du sud",
];
```

Créer un tableau

- On crée un tableau à l'aide d'une paire de **crochets []**
- Tout ce qui se trouve entre les crochets correspond au contenu du tableau.
- Les différents éléments stockés sont séparés par des **virgules ,**

```
let films = [
  "Django Unchained",
  "Interstellar",
  "Green Book : Sur les routes du sud",
];
```

Créer un tableau - types

- Avec JavaScript, on peut stocker dans un tableau des éléments de différents types

```
let tableau = ["bonjour", 7, { message: "Coucou maman" }, true];
```

- Attention: ceci n’est pas le cas de tous les langages de programmation

Nommer un tableau

- Un tableau est destiné à contenir plusieurs éléments
- Une bonne pratique consiste à donner aux variables tableaux des noms exprimant le pluriel, comme par exemple **films**, **acteurs** ou encore **notes**.

```
let films = [...];
let acteurs = [...];
let notes = [...];
```

Taille d’un tableau

- Le nombre d'éléments stockés dans un tableau est appelé sa **taille**

- La taille d'un tableau s'obtient en lui appliquant la propriété **length**.

```
let films = ["Django Unchained", "Interstellar", "Green Book"];

console.log(films.length); // 3
```

- `length = 0` dans le cas d'un tableau vide

Accéder aux éléments

- Chaque élément présent dans un tableau est identifié par un numéro, appelé son **indice** (***index*** en anglais).
- On peut représenter graphiquement un tableau comme un ensemble de cases, chacune stockant une valeur spécifique et associée à un indice.

- Voici comment on pourrait représenter le tableau **films**.

Indice	0	1	2
Valeur	Django Unchained	Interstellar	Green Book

Accéder aux éléments

- L'accès à un élément s'effectue en plaçant cet indice entre crochets

```
let films = ["Django Unchained", "Interstellar", "Green Book"];

console.log(films[0]); // "Django Unchained"
console.log(films[1]); // "Interstellar"
console.log(films[2]); // "Green Book"
```

Accéder aux éléments

- L'indice du premier élément d'un tableau est **0** (~~et non pas 1~~)
- Le plus grand indice utilisable est donc égal à la **taille du tableau - 1**

```
let films = ["Django Unchained", "Interstellar", "Green Book"];

// premier élément
console.log(films[0]); // "Django Unchained"

// dernier élément
console.log(films[films.length - 1]); // "Green Book"
```

Accéder aux éléments

- Si vous utilisez un indice invalide, JS renvoie la valeur spéciale **undefined**

```
let films = ["Django Unchained", "Interstellar", "Green Book"];

console.log(films[3]); // undefined
```

- Dans d'autres langages, cela provoque une erreur : à éviter !

Parcourir un tableau – for

- Avec la boucle **for**, on fait varier l'indice du tableau
  - de 0 (indice du premier élément)
  - à taille du tableau - 1 (indice du dernier)
- pour accéder aux éléments les uns après les autres.

```
let films = ["Django Unchained", "Interstellar", "Green Book"];

for (let i = 0; i < films.length; i++) {
  console.log(films[i]);
}
```

Parcourir un tableau – forEach

- La méthode **forEach()** permet d'appliquer une fonction (anonyme) sur chaque élément du tableau
- Chaque élément du tableau est successivement passé en paramètre à la fonction

```
let films = ["Django Unchained", "Interstellar", "Green Book"];

films.forEach((film) => {
  console.log(film);
});
```

Parcourir un tableau – for-of

- **for-of** a été ajouté récemment dans langage JavaScript
- C’est une version simplifiée de forEach

```
let films = ["Django Unchained", "Interstellar", "Green Book"];

for (let film of films) {
  console.log(film);
}
```

Ajouter des éléments

- La méthode «traditionnelle» pour ajouter un élément est de lui assigner une valeur sur un indice non utilisé

```
let films = ["Django Unchained", "Interstellar", "Green Book"];

films[films.length] = "Les Bronzés";
```

Ajouter des éléments

- La méthode moderne se fait avec la méthode **`push()`**
- Elle prend en paramètre l'élément à insérer, qui est ajouté à la fin du tableau

```
let films = ["Django Unchained", "Interstellar", "Green Book"];

films.push("Les Bronzés"); // Ajoute le film à la fin du tableau

console.log(films[3]); // "Les Bronzés"
```

Ajouter des éléments

- **`unshift()`** permet d'ajouter l'élément au début du tableau

```
let films = ["Django Unchained", "Interstellar", "Green Book"];

films.unshift("Les Bronzés"); // Ajoute le film au début du tableau

console.log(films[0]); // "Les Bronzés"
```

Supprimer des éléments

- **push()** permet d'ajouter l'élément à la fin du tableau
- **pop()** permet de supprimer l'élément à la fin du tableau

```
let films = ["Django Unchained", "Interstellar", "Green Book"];

let film = films.pop(); // Supprime le dernier élément

console.log(films.length); // 2
console.log(film); // "Green Book"
```

- pop permet de récupérer la valeur supprimée

Supprimer des éléments

- **`unshift()`** permet d'ajouter l'élément au début du tableau
- **`shift()`** permet de supprimer l'élément au début du tableau

```
let films = ["Django Unchained", "Interstellar", "Green Book"];

let film = films.shift(); // Supprime et le premier élément

console.log(films.length); // 2
console.log(film); // "Django Unchained"
```

- Tout comme pop, `shift` renvoie la valeur supprimée

Supprimer des éléments

- **`splice()`** permet de supprimer un ou plusieurs éléments d'un coup
  - son premier paramètre est l'indice à partir duquel supprimer
  - et le second est le nombre d'éléments à supprimer.

```
let films = ["Django Unchained", "Interstellar", "Green Book"];

films.splice(1, 1);

console.log(films.length); // 2
console.log(films[0]); // "Django Unchained"
console.log(films[1]); // "Green Book"
```

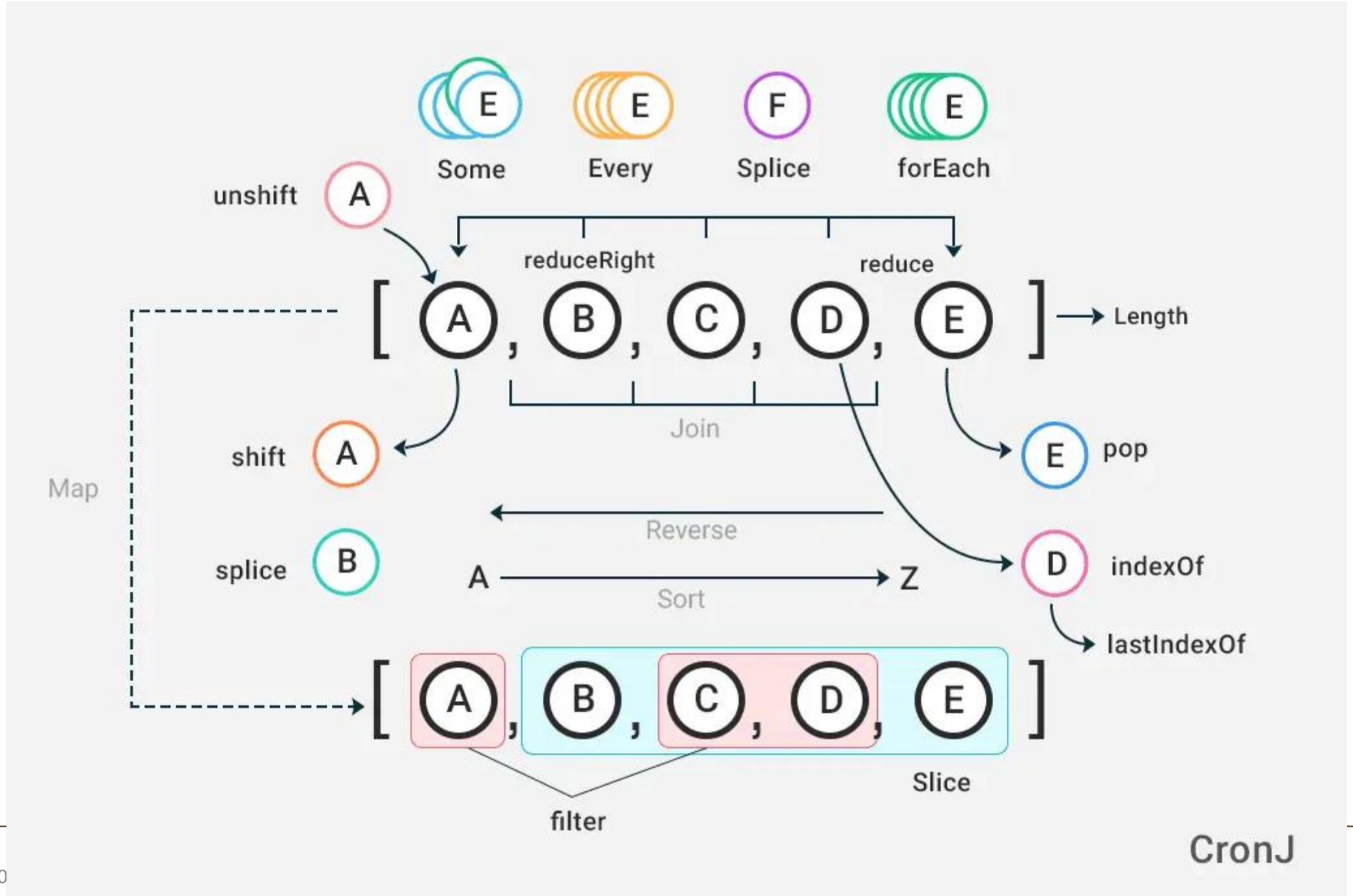
Supprimer des éléments

- Si on ajoute un troisième paramètre à la méthode `splice()`
- Les valeurs supprimées seront remplacées par le troisième paramètre.

```
let films = ["Django Unchained", "Interstellar", "Green Book"];

films.splice(2, 1, "Jungle Book");

console.log(films.length); // 3
console.log(films[2]); // "Jungle Book"
```

Quelques méthodes intéressantes

- concat() Retourne la concaténation de 2 tableaux
- every() Teste si l'ensemble des éléments du tableau répond à la condition de la fonction
- filter() Retourne un tableau contenant les éléments qui vérifient la fonction
- from() Construit un tableau à partir d'un autre objet
- includes() Indique si l'élément passé en paramètre appartient bien au tableau
- indexOf() Retourne l'index du tableau contenant l'élément recherché
- isArray() Retourne true si l'objet passé en paramètre est un tableau
- join() Retourne une chaîne composée de tous les éléments du tableau
- reverse() Inverse l'ordre des éléments du tableau
- some() Teste si au moins un des éléments du tableau répond à la condition
- sort() Trie les éléments de tableau par ordre croissant ou selon la fonction optionnelle


E
E
F
E
Some
Every
Splice
forEach
unshift
A
reduceRight
reduce
[A , B , C , D , E]
Length
Join
shift
A
E
pop
Map
Reverse
splice
B
A
Z
Sort
D
indexOf
lastIndexOf
[A , B , C , D , E]
Slice
filter

CronJ

Exercices

- Exercices 8.x : **Tableaux**